P-Lipsum
A ‘lorem ipsum’ paragraph generator
in plain \TeX\ for plain \TeX\ers.

Sergio Spina
sergio.am.spina@gmail.com

May 8, 2013

Contents

I  Introduction 3

II  The program 4
  1  Usage. 4

III  Macros 5
  2 \lipsum 5
  3 \everystartplipsumpar \everyendplipsumpar 5
  4 \nopar. 6

IV  Thanks 7

V  The code 8

VI  An example 15

VII  The development tree of the code file 16

VIII Indexes 17
5 Chunks.  
6 Identifiers.
Part I
Introduction

I wrote this format for my needs, taking inspiration from \texttt{lipsum.dtx} by Patrick Happel and \texttt{kantlipsum.dtx} by Enrico Gregorio. But those two packages are not useful for me as far as I use only plain \TeX{} and those packages are written in \LaTeX{} and aimed only to the \LaTeX{} world.

Having used \textit{P-Lipsum} for a while and having found it useful I’ve thought that it can be shared with the \TeX{} community.
Part II
The program

I used, to build this package, real phrases of real latin, taken from the Cicero’s *De finibus bonorum et malorum*. There are two main advantages in using real latin: first, it ‘sounds’ better; secondly, in the phrases there are a lot of ‘fi’, ‘fl’, ‘ffi’ and ‘flfl’ groups, useful to see the effects of typographic ligatures with the font choosen for your document.

1 Usage.

To load the format specify \input plipsum into your document.
Part III

Macros

2 \lipsum

It’s the main macro. It can accept one or two arguments separated by an hyphen and an optional parameter.

The argument is one or two different numbers in the range 1-100.

The optional parameter can be one of the following:

\[ s \mid \text{[short]} \]
\[ m \mid \text{[medium]} \]
\[ l \mid \text{[long]} \]

For ex.:

\lipsum{33} \quad \leftarrow \quad \text{A. one argument}
\lipsum[long]{33} \quad \leftarrow \quad \text{B. one optional parameter and one argument}
\lipsum{13-33} \quad \leftarrow \quad \text{C. two arguments}
\lipsum[s]{13-33} \quad \leftarrow \quad \text{D. one optional parameter and two arguments}

In the case ‘A’: it will be typeset just the paragraph of medium length number 33.
In the case ‘B’: it will be typeset the paragraph of long length number 33.
In the case ‘C’: it will be typeset 21 paragraphs of medium length, from the number 13 (included) up to the number 33 (included).
In the case ‘D’: it will be typeset 21 paragraphs of short length, from the number 13 (included) up to the number 33 (included) \(^1\).

Arguments cannot be greater than 100.
It’s ok if the first argument is greater than the second.

\lipsum{33-13} \quad \leftarrow \quad \text{E. same as example ‘C’; it works as well.}

3 \everystartplipsumpar \everystopplipsumpar

These two token lists were in existence in the previous version of P-Lipsum. As far as they can be easily replaced with normal \TeX{} programming tools they have been deleted.

\(^1\)The interface of the \lipsum{} command is slightly different from the previous version. It is not a good practice to change the interface from one version to another, but \textit{this specific package} is not aimed to typeset documents whose destin is to be archived, so the backward compatibility is not an important issue.
4 \nopar.

It eliminates the \par between one paragraph and the following one. This way many paragraphs may become one single big paragraph.

\nopar\lipsum{14-16}\par ← One single paragraph made with all the medium-length paragraphs number 14, number 15 and number 16.
Part IV

Thanks

The ‘lorem ipsum’ paragraphs were generated with the help of

http://loripsum.net

maybe the best lipsum generator on the web.
The main source for the paragraphs in lipsum is the Cicero’s *De finibus bonorum et malorum*. You can find the complete, original text at:

http://www.thelatinlibrary.com/cicero/fin.shtml

I wrote the package with *noweb* by Norman Ramsey, a simple tool for Literate Programming.

http://www.cs.tufts.edu/~nr/noweb/
http://www.ctan.org/tex-archive/web/noweb

I feel much more comfortable with this tool than the \LaTeX\ packages *Doc* and *Docstrip*. 
Part V
The code

Let's begin with the introductory things.

Version informations in a comfortable place.

\[\langle \text{preliminaries} \rangle \equiv \]
\(\begin{align*}
\text{\def\PLversion}{4} \\
\text{\def\PLrevision}{3} \\
\text{\def\PLrevisiondate}{2013/05/08}
\end{align*}\]

The sign '@' for the private macros.

\[\langle \text{preliminaries} \rangle \equiv \]
\(\begin{align*}
\text{\chardef\beforeplipsumatcatcode=\the\catcode'@} \\
\text{\catcode'@=11}
\end{align*}\]

Messages in the log file. Handle the newline with the pipe char.

\[\langle \text{preliminaries} \rangle \equiv \]
\(\begin{align*}
\text{\def\@message#1{{\newlinechar'\l^J\message{#1}}} \\
\text{\def\@errmessage#1{{\newlinechar'\l^J\errmessage{#1}}}}
\end{align*}\]

The first cry of this \TeX\ child.

\[\langle \text{collection} \rangle \equiv \]
\(\begin{align*}
\text{\newcount\c@parnumber} \\
\text{\c@parnumber=0} \\
\text{\def\create@par{{\ifnum\c@parnumber<100 \else \expandafter\def\csname plips@\roman{\the\c@parnumber}\endcsname\fi}}}
\end{align*}\)

Every paragraph is constituted by a macro who’s name can be reached at expansion time via a sequential number, for ex.: \texttt{\lips@xxv{...}, \lips@xxvi{...}, etc.}

Macros from \texttt{\lips@i} up to \texttt{\lips@c} (the first 100 paragraphs) are long paragraphs; macros from \texttt{\lips@ci} up to \texttt{\lips@cc} are medium-length paragraphs and the macros from \texttt{\lips@cci} up to \texttt{\lips@ccc} are short paragraphs.

As far as I’m too lazy to write 300 macro’s names organized this way, the job will be done for me by the subsequent macro, thanks to the magic of \texttt{\csname name... \endcsname}.

\[\langle \text{collection} \rangle \equiv \]
\(\begin{align*}
\text{\newcount\c@parnumber} \\
\text{\c@parnumber=0} \\
\text{\def\create@par{{\ifnum\c@parnumber<100 \else \expandafter\def\csname plips@\roman{\the\c@parnumber}\endcsname\fi}}}
\end{align*}\)
The last touch: I write an equivalent of the primitive \par so I can safely ‘deactivate’ it.

\def\par{}\par

It follows the macro to deactivate \par

\def\nopar{\let\par\space}

Defines:
\nopar, used in chunk 15b.

Now we can define the catalog of Latin phrases containing the characteristics built up till now. We won’t report all the catalog: it’s simply a list of 300 identical macros where the expansion is constituted by a Latin paragraph. So it follow just the first paragraph.

% LONG paragraphs
\create@par{Quid enim necesse est, tamquam meretricem in matronarum coetum, sic voluptatem in virtutum concilium adducere? Nunc dicam de voluptate, nihil scilicet novi, ea tamen, quae te ipsum probaturum esse confidam. Iam quae corporis sunt, ea nec auctoritatem cum animi partibus, comparandam et cognitionem habent faciliorem. Si qua in iis corrugere voluit, deteriora fecit. Polemoni et iam ante Aristoteli ea prima visa sunt, quae paulo ante dixi. Neque solum ea communia, verum etiam paria esse dixerunt. Non enim quae quid verum, sed quid cuique dicendum sit. Levatio igitur vitiorum magna fit in iis, qui habent ad virtutem progressionis aliquantum. A primo, ut opinor, animantium ortu petitur origo summi boni. Nam si quae sunt aliae, falsum est omnis animi voluptates esse e corporis societate. Quod est, ut dixi, habere ea, quae secundum naturam sint, vel omnia vel plurima et maxima. Praetereo multos, in bis doctum hominem et suavem, Hieronymum, quem iam cur Peripateticum appellem nescio.\par}

Anyway, after so much work the format launch an echo of proudness in the .log file.

\message{^^JP-lipsum: created \number\c@parnumber\space paragraphs.^^J}

Before of the beginning of the interface building, in which the users will insert values, let’s define some error messages.

\newhelp\optparams@error{%
}\newhelp\paramexcess@error{The best possible value is 100}
Once builted the foundations we can build the main macro of the format. \lipsum has a lot of behaviors:

1. if it is called with a single parameter (specifically a \langle number\rangle), it expand to a single medium-length paragraph, namely the one that come at the \langle number \rangle + 100\rangleth place in the list of paragraphs.

2. If it is called with two parameters \langle numbers\rangle separated by an hyphen it expands to the all the medium-length paragraphs having the number from the lower parameter to the higher (plus 100), all the between included.

3. It can be called with an optional parameter, alternatively \texttt{s/short} or \texttt{m/medium} or \texttt{l/long}. If the optional parameter is \texttt{s} or \texttt{short} the macro expands to the short paragraphs having the number from the lower parameter to the higher, all the between included. If the optional parameter is \texttt{m} or \texttt{medium} the macro expands to the medium paragraphs having the number from the lower parameter to the higher, all the between included. So the same for the optional parameters \texttt{l} or \texttt{long} \langle number \rangle + 200).

So the details of the macro are simple (well, more or less): first the macro look for a square bracket into the parameter:

\begin{verbatim}
\if\[\firsttak\verbatim\optpar\\
\else\mediumtrue\\
\let\next\noperpar\\
\fi
\next\def\lipsum{\futurelet\firsttak\lipsum}
\end{verbatim}

If a square bracket is found then is called the macro \verbatim\optpar, otherwise \verbatim\noperpar. If no square bracket is found it follows that the latin paragraphs have to be medium-length.

\begin{verbatim}
\if\firsttak\verbatim\optpar\\
\else\mediumtrue\\
\let\next\noperpar\fi
\next\def\lipsum{\futurelet\firsttak\lipsum}
\end{verbatim}
If a square bracket is found the only thing to do is to set the various boolean values. As the macro `opt@par` consumes the optional parameter, then it call the servant macro as if it would have been called without optional parameters at all.

```
\def\opt@par[#1]#2{\def\param@ne{#1}\
  \def\@l@{l}\def\@long@{long}\
  \def\@m@{m}\def\@medium@{medium}\
  \def\@s@{s}\def\@short@{short}\
  \ifx\param@ne\@l@\shortfalse\mediumfalse\longtrue \else\
  \ifx\param@ne\@long@\shortfalse\mediumfalse\longtrue \else\
  \ifx\param@ne\@m@\shortfalse\mediumtrue\longfalse \else\
  \ifx\param@ne\@medium@\shortfalse\mediumtrue\longfalse \else\
  \ifx\param@ne\@short@\shorttrue\mediumfalse\longfalse \else\
  \errhelp\optparams@error\
  \@errmessage{^^JP-lipsum: !! ERROR !!\n  Wrong optional parameter.^^J}
  \fi\fi\fi\fi\fi\fi
\no@opt@par{#2}}
```

Now it is necessary to scan the parameter to ensure that there is an hyphen. In this case is called the macro `\noopt@parA`, otherwise the macro `\noopt@parB`

```
\newif\ifhyphen
\def\noopt@par#1{\scan#1-;\end
  \ifhyphen\noopt@parA#1\end\else\noopt@parB#1\end\fi}
```
\scan look for an hyphen into the argument of \lipsum. The technique is simple: \lipsum calls \scan referring to its own parameter and adding an hyphen and a semicolon. This way \scan can be called — accordingly to the parameter of \lipsum — alternatively in one of these forms:

1. \scan<par1>-;\end
2. \scan<par1>-<par2>-;\end

where the hyphens and the \end are delimiters of \scan. Now the problem is solved: if the second parameter of \scan is a semicolon the parameter passed to \lipsum has not hyphens and it’s a single (number). Otherwise the parameter passed to \lipsum contains an hyphen and is thus constituted by two (numbers).

The code is much simpler than the explanation.

\begin{verbatim}
\def\scan#1-#2\end{\ifx;#2\hyphenfalse\else\hyphentrue\fi}
\end{verbatim}
Now everything is simple and the macros are self-explaining.
If the parameter of \lipsum has an hyphen will be performed \noopt@parA;
onetheless \noopt@parB.

\[(\text{interface 9e}) + \equiv (16) \triangleq 12\]

\begin{verbatim}
\newcount\c@plipsumAone \newcount\c@plipsumAtwo \newcount\c@max
\c@max\c@parnumber \divide\c@max by3\relax
\def\noopt@parA#1-#2\end{% 
  \global\c@plipsumAone \global\c@plipsumAtwo \global\c@max 
  \errhelp\paramexcess@error \@errmessage{``JP-lipsum: ERROR at line `\the\inputlineno. The first parameter of `\string\lipsum\space is too big.``}fi 
  \ifnum\c@plipsumAtwo\c@max 
    \errhelp\paramexcess@error \@errmessage{``JP-lipsum: ERROR at line `\the\inputlineno. The second parameter of `\string\lipsum\space is too big.``}fi
  \ifmedium
    \advance\c@plipsumAone by100\relax
    \advance\c@plipsumAtwo by100\relax
  \else\ifshort
    \advance\c@plipsumAone by200\relax
    \advance\c@plipsumAtwo by200\relax
  \fi\fi
  \ifnum\c@plipsumAone\c@plipsumAtwo
    \count@=\c@plipsumAone
    \c@plipsumAone=\c@plipsumAtwo
    \c@plipsumAtwo=\count@\fi
  \types@t}}
\def\noopt@parB#1\end{% 
  \c@plipsumAone#1
  \ifnum\c@plipsumAone\c@max 
    \errhelp\paramexcess@error \@errmessage{``JP-lipsum: ERROR at line `\the\inputlineno. The parameter of `\string\lipsum\space is too big.``}fi
  \ifmedium\advance\c@plipsumAone by100\relax
  \else\ifshort\advance\c@plipsumAone by200\relax\fi\fi
  \csname plips@\romannumeral\c@plipsumAone\endcsname}
\def\types@t{%\let\next\relax
\end{verbatim}
May 8, 2013

\ifnum\c@plipsumAone>\c@plipsumAtwo\else
\csname plips@\romannumeral\the\c@plipsumAone\endcsname
\advance\c@plipsumAone by1\relax
\let\next\types@t\fi
\next\}

That’s all, folks.

$\langle\text{ending 14}\rangle\equiv (16)\$

$\text{catcode}'@=\text{beforeplipsumatcatcode}$

\endinput
Part VI
An example

The format comes with a simple example (pliptest.tex) aimed to show the functionalities of the macros.

\begin{verbatim}
\input plipsum
\end{verbatim}

\begin{verbatim}
\lipsum[short]{13}
\end{verbatim}

\begin{verbatim}
\lipsum[medium]{13}
\end{verbatim}

\begin{verbatim}
\lipsum[long]{13}
\end{verbatim}

\begin{verbatim}
\lipsum[short-15]
\end{verbatim}

\begin{verbatim}
\lipsum[medium-15]
\end{verbatim}

\begin{verbatim}
\lipsum[long-15]
\end{verbatim}

\begin{verbatim}
\lipsum[very-long]{13-31}
\end{verbatim}

\begin{verbatim}
\bye
\end{verbatim}

\texttt{\textbackslash nopar}\texttt{9b.}

\texttt{\textbackslash nopar}\texttt{9b.}
Part VII
The development tree of the code file

\begin{verbatim}
\verb|\langle plipsum.tex 16\rangle| ≡
\verb|\% This is file 'plipsum.tex'|
\verb|\langle banner 15a\rangle|
\verb|\langle preliminaries 8a\rangle|
\verb|\langle collection 8e\rangle|
\verb|\langle paragraphs ??\rangle|
\verb|\langle echo 9a\rangle|
\verb|\langle interface 9e\rangle|
\verb|\langle ending 14\rangle|
\verb|\% end of file 'plipsum.tex'|
\end{verbatim}
Part VIII
Indexes

5 Chunks.

\begin{quote}
\textit{banner 15a}
\textit{collection 8e}
\textit{echo 9d}
\textit{ending 14}
\textit{interface 9e}
\textit{plipsum.tex 16}
\textit{pliptest.tex 15b}
\textit{preliminaries 8a}
\end{quote}

6 Identifiers.

\texttt{\textbackslash nopar: 9b, 15b}